Introduction to the **tm** Package
Text Mining in R

Ingo Feinerer

February 7, 2008

Abstract

This vignette gives a short overview over available features in the **tm** package for text mining purposes in R.

Loading the Package

Before actually working we need to load the package:

```
> library("tm")
```

Data Import

The main structure for managing documents is a so-called text document collection, denoted as corpus in linguistics (Corpus). Its constructor takes following arguments:

- `object`: a Source object which abstracts the input location.
- `readerControl`: a list with the named components `reader`, `language`, and `load`. A reader constructs a text document from a single element delivered by a source. A reader must have the argument signature `(elem, load, language, id)`. The first argument is the element provided from the source, the second gives the text's language, the third indicates whether the user wants to load the documents immediately into memory, and the fourth is a unique identification string. If the passed over `reader` object is of class FunctionGenerator, it is assumed to be a function generating a reader. This way custom readers taking various parameters (specified in `...`) can be built, which in fact must produce a valid reader signature but can access additional parameters via lexical scoping (i.e., by the including environment).
- `dbControl`: a list with the named components `useDb` indicating that database support should be activated, `dbName` giving the filename holding the sourced out objects (i.e., the database), and `dbType` holding a valid database type as supported by `filehash`. Under activated database support the `tm` packages tries to keep as few as possible resources in memory under usage of the database.

- ...: Further arguments to the reader.

Available sources are DirSource, CSVSource, GmaneSource and ReutersSource which handle a directory, a mixed CSV, a Gmane mailing list archive Rss feed or a mixed Reuters file (mixed means several documents are in a single file). Except DirSource, which is designated solely for directories on a file system, all other implemented sources can take connections as input (a character string is interpreted as file path).

This package ships with several readers (`readPlain()` (default), `readRCV1()`, `readReut21578XML()`, `readGmane()`, `readNewsgroup()`, `readPDF()`, `readDOC()` and `readHTML()`). Each source has a default reader which can be overridden. E.g., for `DirSource` the default just reads in the whole input files and interprets their content as text.

Plain text files in a directory:

```
> txt <- system.file("texts", "txt", package = "tm")
> (ovid <- Corpus(DirSource(txt),
+                     readerControl = list(reader = readPlain,
+                                          language = "la",
+                                          load = TRUE)))

A text document collection with 5 text documents
```

A single comma separated values file:

```
> cars <- system.file("texts", "cars.csv", package = "tm")
> Corpus(CSVSource(cars))

A text document collection with 4 text documents
```

Reuters21578 files either in directory (one document per file) or a single file (several documents per file). Note that connections can be used as input:

```
> # Reuters21578 XML
> reut21578 <- system.file("texts", "reut21578", package = "tm")
> reut21578XML <- system.file("texts", "reut21578.xml", package = "tm")
> reut21578XMLgz <- system.file("texts", "reut21578.xml.gz", package = "tm")
> (reut21578TDC <- Corpus(DirSource(reut21578),
+                             readerControl = list(reader = readReut21578XML,
+                                                  language = "en_US",
+                                                  load = FALSE)))

A text document collection with 10 text documents

> Corpus(ReutersSource(reut21578XML),
+            readerControl = list(reader = readReut21578XML,
+                                 language = "en_US", load = FALSE))

A text document collection with 10 text documents

> Corpus(ReutersSource(gzfile(reut21578XMLgz)),
+            readerControl = list(reader = readReut21578XML,
+                                 language = "en_US", load = FALSE))
```

```
A text document collection with 10 text documents
```

Depending on your exact input format you might find `preprocessReut21578XML()` useful. For the original downloadable archive this function can correct invalid UTF8 encodings and can copy each text document into a separate file to enable load on demand.

Analogously we can construct collections for files in the Reuters Corpus Volume 1 format:

```
> rcv1 <- system.file("texts", "rcv1", package = "tm")
> rcv1XML <- system.file("texts", "rcv1.xml", package = "tm")
> Corpus(DirSource(rcv1), readerControl = list(reader = readRCV1,
+     language = "en_US", load = TRUE))

A text document collection with 2 text documents

> Corpus(ReutersSource(rcv1XML), readerControl = list(reader = readRCV1,
+     language = "en_US", load = FALSE))

A text document collection with 2 text documents
```

Or mails from newsgroups (as found in the UCI KDD newsgroup data set):

```
> newsgroup <- system.file("texts", "newsgroup", package = "tm")
> Corpus(DirSource(newsgroup), readerControl = list(reader = readNewsgroup,
+     language = "en_US", load = TRUE))

A text document collection with 6 text documents
```

RSS feed as delivered by Gmane for the R mailing list archive:

```
> rss <- system.file("texts", "gmane.comp.lang.r.gr.rdf", package = "tm")
> Corpus(GmaneSource(rss), readerControl = list(reader = readGmane,
+     language = "en_US", load = FALSE))

A text document collection with 21 text documents
```

For very simple HTML documents:

```
> html <- system.file("texts", "html", package = "tm")
> Corpus(DirSource(html), readerControl = list(reader = readHTML,
+     load = TRUE))

A text document collection with 1 text document
```

And for PDF documents:

```
> pdf <- system.file("texts", "pdf", package = "tm")
> Corpus(DirSource(pdf), readerControl = list(reader = readPDF,
+     language = "en_US", load = TRUE))

A text document collection with 1 text document
```

Note that `readPDF()` needs `pdftotext` and `pdfinfo` installed on your system to be able to extract the text and meta information from your PDFs.

Finally, for MS Word documents there is the reader function `readDOC()`. You need `antiword` installed on your system to be able to extract the text from your Word documents.

Data Export

For the case you have created a text collection via manipulating other objects in R, thus do not have the texts already stored, and want to save the text documents to disk, you can simply use standard R routines for writing out plain text documents. E.g.,

```
> lapply(ovid, function(x) writeLines(x, paste(ID(x), ".txt",
+     sep = "")))
```

Alternatively there is the function `writeCorpus()` which encapsulates this functionality.

Inspecting the Text Document Collection

Custom `show` and `summary` methods are available, which hide the raw amount of information (consider a collection could consists of several thousand documents, like a database). `summary` gives more details on metadata than `show`, whereas in order to actually see the content of text documents use the command `inspect` on a collection.

```
> show(ovid)

A text document collection with 5 text documents

> summary(ovid)

A text document collection with 5 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

> inspect(ovid[1:2])

A text document collection with 2 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

[[1]]
 [1] "    Si quis in hoc artem populo non novit amandi,"
 [2] "         hoc legat et lecto carmine doctus amet."
 [3] "    arte citae veloque rates remoque moventur,"
 [4] "         arte leves currus: arte regendus amor."
 [5] ""
```

```
 [6] "    curribus Automedon lentisque erat aptus habenis,"
 [7] "         Tiphys in Haemonia puppe magister erat:"
 [8] "    me Venus artificem tenero praefecit Amori;"
 [9] "         Tiphys et Automedon dicar Amoris ego."
[10] "    ille quidem ferus est et qui mihi saepe repugnet:"
[11] ""
[12] "         sed puer est, aetas mollis et apta regi."
[13] "    Phillyrides puerum cithara perfecit Achillem,"
[14] "         atque animos placida contudit arte feros."
[15] "    qui totiens socios, totiens exterruit hostes,"
[16] "         creditur annosum pertimuisse senem."

[[2]]
 [1] "    quas Hector sensurus erat, poscente magistro"
 [2] "         verberibus iussas praebuit ille manus."
 [3] "    Aeacidae Chiron, ego sum praeceptor Amoris:"
 [4] "         saevus uterque puer, natus uterque dea."
 [5] "    sed tamen et tauri cervix oneratur aratro,"
 [6] ""
 [7] "         frenaque magnanimi dente teruntur equi;"
 [8] "    et mihi cedet Amor, quamvis mea vulneret arcu"
 [9] "         pectora, iactatas excutiatque faces."
[10] "    quo me fixit Amor, quo me violentius ussit,"
[11] "         hoc melior facti vulneris ultor ero:"
[12] ""
[13] "    non ego, Phoebe, datas a te mihi mentiar artes,"
[14] "         nec nos aëriae voce monemur avis,"
[15] "    nec mihi sunt visae Clio Cliusque sorores"
[16] "         servanti pecudes vallibus, Ascra, tuis:"
[17] "    usus opus movet hoc: vati parete perito;"
```

Transformations

Once we have a text document collection one typically wants to modify the documents in it, e.g., stemming, stopword removal, et cetera. In **tm**, all this functionality is subsumed into the concept of *transformation*s. Transformations are done via the `tmMap` function which applies a function to all elements of the collection. Basically, all transformations work on single text documents and `tmMap` just applies them to all documents in a document collection.

Loading Documents into Memory

If the source objects supports load on demand, but the user has not enforced the package to load the input content directly into memory, this can be done manually via `loadDoc`. Normally it is not necessary to call this explicitly, as other functions working on text corpora trigger this function for not-loaded documents (the corpus is automatically loaded if accessed via `[[`).

```
> reut21578TDC <- tmMap(reut21578TDC, loadDoc)
```

Converting to Plaintext Documents

The text document collection `reut21578TDC` contains documents in XML format. We have no further use for the XML interna and just want to work with the text content. This can be done by converting the documents to plaintext documents. It is done by the generic `asPlain`.

```
> reut21578TDC <- tmMap(reut21578TDC, asPlain)
```

Eliminating Extra Whitespace

Extra whitespace is eliminated by:

```
> reut21578TDC <- tmMap(reut21578TDC, stripWhitespace)
```

Convert to Lower Case

Conversion to lower case by:

```
> reut21578TDC <- tmMap(reut21578TDC, tmTolower)
```

Remove Stopwords

Removal of stopwords by:

```
> reut21578TDC <- tmMap(reut21578TDC, removeWords, stopwords("english"))
```

Stemming

Stemming is done by:

```
> tmMap(reut21578TDC, stemDoc)

A text document collection with 10 text documents
```

Filters

Often it is of special interest to filter out documents satisfying given properties. For this purpose the function `tmFilter` is designated. It is possible to write custom filter functions, but for most cases the default filter does its job: it integrates a minimal query language to filter metadata. Statements in this query language are statements as used for subsetting data frames.

E.g., the following statement filters out those documents having `COMPUTER TERMINAL SYSTEMS <CPML> COMPLETES SALE` as their heading and an `ID` equal to 10 (both are metadata slot variables of the text document).

```
> query <- "identifier == '10' &
+           heading == 'COMPUTER TERMINAL SYSTEMS <CPML> COMPLETES SALE'"
> tmFilter(reut21578TDC, query)

A text document collection with 1 text document
```

There is also a full text search filter available which accepts regular expressions:

```
> tmFilter(reut21578TDC, FUN = searchFullText, "partnership",
+     doclevel = TRUE)

A text document collection with 1 text document
```

Adding Data or Metadata

Text documents or metadata can be added to text document collections with `appendElem` and `appendMeta`, respectively. The text document collection has two types of metadata: one is the metadata on the document collection level (`cmeta`), the other is the metadata related to the individual documents (e.g., clusterings) (`dmeta`) in form of a dataframe. For the method `appendElem` it is possible to give a row of values in the dataframe for the added data element.

```
> data(crude)
> reut21578TDC <- appendElem(reut21578TDC, crude[[1]], 0)
> reut21578TDC <- appendMeta(reut21578TDC, cmeta = list(test = c(1,
+     2, 3)), dmeta = list(cl1 = 1:11))
> summary(reut21578TDC)

A text document collection with 11 text documents

The metadata consists of 3 tag-value pairs and a data frame
Available tags are:
  create_date creator test
Available variables in the data frame are:
  MetaID cl1

> CMetaData(reut21578TDC)

An object of class "MetaDataNode"
Slot "NodeID":
[1] 0

Slot "MetaData":
$create_date
[1] "2008-02-07 12:06:24 CET"

$creator
 LOGNAME
"hornik"

$test
[1] 1 2 3


Slot "children":
list()

> DMetaData(reut21578TDC)
```

```
   MetaID cl1
1       0   1
2       0   2
3       0   3
4       0   4
5       0   5
6       0   6
7       0   7
8       0   8
9       0   9
10      0  10
11      0  11
```

Removing Metadata

The metadata of text document collections can be easily modified or removed:

```
> data(crude)
> reut21578TDC <- removeMeta(reut21578TDC, cname = "test",
+     dname = "cl1")
> CMetaData(reut21578TDC)

An object of class "MetaDataNode"
Slot "NodeID":
[1] 0

Slot "MetaData":
$create_date
[1] "2008-02-07 12:06:24 CET"

$creator
 LOGNAME
"hornik"


Slot "children":
list()

> DMetaData(reut21578TDC)

   MetaID
1       0
2       0
3       0
4       0
5       0
6       0
7       0
8       0
9       0
```

```
10      0
11      0
```

Operators

Many standard operators and functions (`[`, `[<-`, `[[`, `[[<-`, `c`, `length`, `lapply`, `sapply`) are available for text document collections with semantics similar to standard R routines. E.g., `c` concatenates two (or more) text document collections. Applied to several text documents it returns a text document collection. The metadata is automatically updated, if text document collections are concatenated (i.e., merged).

Note also the custom element-of operator—it checks whether a text document is already in a text document collection (metadata is not checked, only the corpus):

```
> crude[[1]] %IN% reut21578TDC

[1] TRUE

> crude[[2]] %IN% reut21578TDC

[1] FALSE
```

Keeping Track of Text Document Collections

There is a mechanism available for managing text document collections. It is called TextRepository. A typical use would be to save different states of a text document collection. A repository has metadata in list format which can be either set with `appendElem` as additional argument (e.g., a date when a new element is added), or directly with `appendMeta`.

```
> data(acq)
> repo <- TextRepository(reut21578TDC)
> repo <- appendElem(repo, acq, list(modified = date()))
> repo <- appendMeta(repo, list(moremeta = 5:10))
> summary(repo)

A text repository with 2 text document collections

The repository metadata consists of 3 tag-value pairs
Available tags are:
created modified moremeta

> RepoMetaData(repo)

$created
[1] "2008-02-07 12:06:25 CET"

$modified
[1] "Thu Feb  7 12:06:25 2008"

$moremeta
[1]  5  6  7  8  9 10
```

```
> summary(repo[[1]])

A text document collection with 11 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

> summary(repo[[2]])

A text document collection with 50 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID
```

Creating Term-Document Matrices

A common approach in text mining is to create a term-document matrix for given texts. In this package the class TermDocMatrix handles sparse matrices for text document collections.

```
> tdm <- TermDocMatrix(reut21578TDC)
> Data(tdm)[1:8, 150:155]

8 x 6 sparse Matrix of class "dgCMatrix"
  weeks year york zone activities america
1     1    2    8    1          .       .
2     .    .    .    .          1       2
3     .    .    .    .          .       .
4     1    1    .    .          .       .
5     .    .    .    .          .       .
6     .    .    .    .          .       .
7     .    .    .    .          .       .
8     .    .    .    .          .       .
```

Operations on Term-Document Matrices

Besides the fact that on the `Data` part of this matrix a huge amount of R functions (like clustering, classifications, etc.) is possible, this package brings some shortcuts. Consider we want to find those terms that occur at least 5 times:

```
> findFreqTerms(tdm, 5, Inf)
```

```
 [1] "bags"        "cocoa"       "comissaria"  "crop"        "dec"
 [6] "dlrs"        "july"        "mln"         "sales"       "sept"
[11] "smith"       "times"       "total"       "york"        "oil"
[16] "analysts"    "bankamerica" "debt"        "price"       "stock"
[21] "the"         "level"       "apr"         "feb"         "mar"
[26] "nil"         "prev"        "computer"    "terminal"
```

Or we want to find associations (i.e., terms which correlate) with at least 0.97 correlation for the term `crop`:

```
> findAssocs(tdm, "crop", 0.97)
```

```
       crop         155         221         325         340
       1.00        0.98        0.98        0.98        0.98
        345         350         351         375         380
       0.98        0.98        0.98        0.98        0.98
        400         415         450         480         750
       0.98        0.98        0.98        0.98        0.98
        753         780         785         850         870
       0.98        0.98        0.98        0.98        0.98
        875         880         995  alleviating       areas
       0.98        0.98        0.98        0.98        0.98
  argentina    arrivals      arroba         aug        bags
       0.98        0.98        0.98        0.98        0.98
      bahia        bean      booked      butter      buyers
       0.98        0.98        0.98        0.98        0.98
       cake     carnival certificates       cocoa  comissaria
       0.98        0.98        0.98        0.98        0.98
consignment   continued   covertible    cruzados  cumulative
       0.98        0.98        0.98        0.98        0.98
  currently         dec   delivered destinations  dificulties
       0.98        0.98        0.98        0.98        0.98
      doubt      doubts     drought         dry         end
       0.98        0.98        0.98        0.98        0.98
  estimated   estimates experiencing   exporters     farmers
       0.98        0.98        0.98        0.98        0.98
      final         fit         fob       hands  harvesting
       0.98        0.98        0.98        0.98        0.98
       held    humidity     hundred   improving    included
       0.98        0.98        0.98        0.98        0.98
       june       kilos      levels      liquor         may
       0.98        0.98        0.98        0.98        0.98
      means      midday   middlemen       named      nearby
       0.98        0.98        0.98        0.98        0.98
     normal   obtaining      period       ports practically
       0.98        0.98        0.98        0.98        0.98
 processors   prospects   published     quality  registered
       0.98        0.98        0.98        0.98        0.98
  reluctant    restored      review        rose     routine
       0.98        0.98        0.98        0.98        0.98
      sales      season     selling        sept    shipment
```

```
      0.98        0.98        0.98        0.98        0.98
  shippers     showers        spot       stage    superior
      0.98        0.98        0.98        0.98        0.98
  temporao    thousand  throughout       times       tonne
      0.98        0.98        0.98        0.98        0.98
     trade     uruguay        view      weekly        york
      0.98        0.98        0.98        0.98        0.98
      zone
      0.98
```

The function also accepts a matrix as first argument (which does not inherit from a term-document matrix). This matrix is then interpreted as a correlation matrix and directly used. With this approach different correlation measures can be employed.

Term-document matrices tend to get very big already for normal sized datasets. Therefore we provide a method to remove *sparse* terms, i.e., terms occurring only in very few documents. Normally, this reduces the matrix dramatically without losing significant relations inherent to the matrix:

```
> removeSparseTerms(tdm, 0.4)

An object of class "TermDocMatrix"
Slot "Data":
11 x 2 sparse Matrix of class "dgTMatrix"
     Terms
      dlrs reuter
  1     14      1
  2      .      1
  3      2      1
  4      3      1
  5      2      1
  6      .      1
  7      1      1
  8      2      1
  9      .      1
  10     4      1
  127    2      1

Slot "Weighting":
[1] "term frequency"
```

This function call removes those terms which have at least a 40 percentage of sparse (i.e., terms occurring 0 times in a document) elements.

Dictionary

A dictionary is a (multi-)set of strings. It is often used to represent relevant terms in text mining. We provide a class Dictionary implementing such a dictionary concept. It can be created via the `Dictionary` constructor, e.g.,

```
> (d <- Dictionary(c("dlrs", "crude", "oil")))
```

```
An object of class "Dictionary"
[1] "dlrs"  "crude" "oil"
```

and may be passed over to the `TermDocMatrix` constructor. Then the created matrix is tabulated against the dictionary, i.e., only terms from the dictionary appear in the matrix. This allows to restrict the dimension of the matrix a priori and to focus on specific terms for distinct text mining contexts, e.g.,

```
> tdmD <- TermDocMatrix(reut21578TDC, list(dictionary = d))
> Data(tdmD)
```

```
11 x 3 sparse Matrix of class "dgCMatrix"
     Terms
      dlrs crude oil
  1     14     .   .
  2      .     .   3
  3      2     .   .
  4      3     .   .
  5      2     .   .
  6      .     .   2
  7      1     .   .
  8      2     .   1
  9      .     .   .
  10     4     .   .
  127    2     2   5
```

You can also create a dictionary from a term-document matrix via `createDictionary` holding all terms from the matrix e.g.,

```
> createDictionary(tdm)[100:110]
```

```
 [1] "midday"    "middlemen" "mln"       "named"     "nearby"
 [6] "normal"    "obtaining" "oct"       "offer"     "period"
[11] "ports"
```